

## REMARKS

### I. Introduction

In response to the Office Action dated November 7, 2002, claims 7-10, 21-38 and 40-57 have been amended. Claims 1-57 remain in the application. Re-examination and re-consideration of the application, as amended, is requested.

### II. Claim Amendments

Applicants' attorney has made amendments to the claims as indicated above. These amendments were made solely for the purpose of clarifying the language of the claims, and were not required for patentability or to distinguish the claims over the prior art.

### III. Non-Art Objections

In paragraphs (3)-(4) of the Office Action, the Office Action stated that claims 21-38 and 40-57 were withdrawn from consideration because they were depending on improper claim numbers. Moreover, claims 42-59 were renumbered as 40-57 because there were no claims 40 and 41.

Applicants' attorney acknowledges the renumbering of the claims and has amended the claims to overcome the objections. Further, Applicants' attorney traverses the "withdrawal" of the claims, and respectfully submits that the Office lacks the authority to make such a withdrawal.

### IV. Prior Art Rejections

#### A. The Office Action Rejections

In paragraphs (5)-(6) of the Office Action, claims 1, 20, and 39 were rejected under 35 U.S.C. §102(b) as being unpatentable over Bhattacharya et al., U.S. Patent No. 5,797,000 (Bhattacharya). In paragraphs (7)-(8) of the Office Action, claims 2-3 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya as applied to claims 1, 20, and 39, and in view of Hintz et al., U.S. Patent No. 5,222,235 (Hintz). In paragraph (9) of the Office Action, claims 4-6 were rejected under 35 U.S.C. §103(a) as being unpatentable over Bhattacharya as applied to claims 1, 20, and 39, and in view of Bordonaro et al., U.S. Patent No. 5,307,485 (Bordonaro). However, in paragraph (11) of the Office Action, claims 12-19 were indicated as being allowable if rewritten in independent form to include the base claim and any intervening claims.

Applicants' attorney acknowledges the indication of allowable claims, but respectfully traverse these rejections.

**B. Applicants' Independent Claims**

Applicants' independent claims 1, 16 and 30 are directed to loading data into a data store connected to a computer. Independent claim 1 is representative and comprises the steps of:

identifying memory constraints;

identifying processing capabilities; and

determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities.

**C. The Bhattacharya Reference**

Bhattacharya describes a method of performing a parallel join operation on a pair of relations R1 and R2 in a system containing P processors organized into Q clusters of P/Q processors each. The system contains disk storage for each cluster, shared by the processors of that cluster, together with a shared intermediate memory (SIM) accessible by all processors. The relations R1 and R2 to be joined are first sorted on the join column. The underlying domain of the join column is then partitioned into P ranges of equal size. Each range is further divided into M subranges of progressively decreasing size to create MP tasks T.sub.m,p, the subranges of a given range being so sized relative to one another that the estimated completion time for task T.sub.m,p is a predetermined fraction that of task T.sub.m-1,p. Tasks T.sub.m,p with larger time estimates are assigned (and the corresponding tuples shipped) to the cluster to which processor p belongs, while tasks with smaller time estimates are assigned to the SIM, which is regarded as a universal cluster (cluster 0). The actual task-to-processor assignments are determined dynamically during the join phase in accordance with the dynamic longest processing time first (DLPT) algorithm. Each processor within a cluster picks its next task at any given decision point to be the one with the largest time estimate which is owned by that cluster or by cluster 0.

**D. The Hintz Reference**

Hintz describes a reorganization method of DB2 data files exploring parallel processing, and asynchronous I/O to a great extent. It includes means to estimate an optimum configuration of system resources, such as storage devices (DASD devices), memory, and CPUs, etc, during

reorganizations. The method mainly consists of four components, (1) concurrent indexing, (2) concurrent unloading of data file partitions, (3) efficient reloading of DB2 data pages and DB2 space maps, and (4) means to reduce access constraints to the DB2 recovery table.

E. The Bordonaro Reference

Bordonaro describes a system and method for merging a plurality of sorted lists using multiple processors having access to a common memory in which N sorted lists which may exceed the capacity of the common memory are merged in a parallel environment. Sorted lists from a storage device are loaded into common memory and are divided into a number of tasks equal to the number of available processors. The records assigned to each task are separately sorted, and used to form a single sorted list. A multi-processing environment takes advantage of its organization during the creation of the tasks, as well as during the actual sorting of the tasks.

F. Applicants' Claimed Invention Is Patentable Over The References

Applicants' attorney respectfully submits that Applicants' claimed invention is patentable over the references. Specifically, Applicants' attorney asserts that the references do not teach or suggest the limitations recited in Applicants' independent claims 1, 20 and 39.

The Office Action states that, as per claims 1, 20, and 39, Bhattacharya et al. disclose a method of loading data into a data store connected to a computer, the method comprising the steps of: identifying processing capabilities (fig. 1, number of processors p, col. 4, lines 41 - 63); and determining a number of load (col. 2, lines 57 - 65, col. 11, lines 44 - 53), and sort processes (col. 2, line 66 - col. 3, line 6) to be started in parallel based on the identified memory constraints and processing capabilities (col. 1, lines 24 - 28, parallel tasks based on processing capabilities: col. 4, line 64 - col. 5, line 8, parallel sort processing: col. 2, lines 66 - col. 3, line 6).

Further, the Office Action states that, although Bhattacharya et al. did not specifically indicate that the number of load and sort processes to be started in parallel be determined on the constraints of memory and processing capability limitations, nevertheless, the fact that loading and sorting processes that Bhattacharya et al. show to be started in parallel, with the description of the memory constraints and the number of processors (as indicated above), inherently meant the limitation (number of load and sort tasks that can be processed) is based on the memory and processing capabilities/constraints.

Applicants' attorney disagrees. The cited portions of Bhattacharya do not teach or suggest the limitation "determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities,"

For example, the cited portions are set forth below:

Col. 2, lines 57 – 65

The invention described here relates to a parallel query method which has a high likelihood of balancing the load well in the face of either initial load imbalance due to data skew or later load imbalance due to stochastic process variations. The invention is described in the context of a sort merge join. However, the same basic method can also be applied to hash joins, sorts, or other queries in a natural manner.

Col. 11, lines 44 – 53

7. The QP determines, using mechanisms that do not form part of this invention, how many partitions T.sub.j should be split into, and how to describe the partition scope (step 1212). Such mechanisms might include a predetermined number from the QP or database instance catalogs, the number of working processing units, the number of database instances available, or a computation based on current system load across all of the processing units or database instances. Once selected, the number of partitions for T.sub.j is denoted as M.sub.j. (M.sub.j corresponds to MP in the discussion further above.)

Col. 2, line 66 - col. 3, line 6

In a parallel sort merge join, the relations to be joined are first sorted, in parallel, within their clusters 102 (FIG. 1). In a naive parallel sort merge join, the underlying join column domain might be partitioned into P ranges of equal size, and the tuples transferred accordingly among the clusters 102. However, given a nonuniform distribution of tuples across the underlying domain, there is no guarantee that the amount of join phase work will be equal.

Col. 1, lines 24 – 28

This invention relates generally to a method of performing a parallel query in a multiprocessor environment and, more particularly, to a method for performing such a query with load balancing in an environment with shared disk clusters, shared intermediate memory or both.

Col. 4, line 64 - col. 5, line 8

Processors 104 are used for the concurrent parallel execution of tasks making up database queries, as described below. A query may originate either from one of the processors 104 or from a separate front-end query processor as described in the concurrently filed application of T. Borden et al., Ser. No. 08/148,091, now U.S. Pat. No. 5,495,606. As further described in that application, within each cluster 102 the query splitting and scheduling steps described below may be performed by an additional processor or processors (not shown) similar to processors 104; such additional processors would not be counted among the P/Q processors 104 per complex 102 to which tasks are assigned.

For example, col. 2, lines 57 – 65 of Bhattacharya relates to “balancing the load,” in the context of performing a sort merge join, but says nothing about “determining a number of load ... processes to be started in parallel based on the identified memory constraints and processing capabilities.”

Similarly, col. 11, lines 44 – 53 of Bhattacharya relates to “how to describe the partition scope” using “a computation based on current system load,” but says nothing about “determining a number of load ... processes to be started in parallel based on the identified memory constraints and processing capabilities.”

In another example, col. 2, line 66 - col. 3, line 6 of Bhattacharya relates to “a parallel sort merge join,” where “the relations to be joined are first sorted,” but says nothing about “determining a number of sort ... processes to be started in parallel based on the identified memory constraints and processing capabilities.”

In yet another example, col. 1, lines 24 – 28 of Bhattacharya relates to “performing a parallel query in a multiprocessor environment,” and “performing such a query with load balancing,” but says nothing about “determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities.”

Finally, in still another example, col. 4, line 64 - col. 5, line 8 of Bhattacharya relates to “concurrent parallel execution of tasks making up database queries,” but says nothing about “determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities.”

Consequently, it cannot be said that Bhattacharya teaches or suggests, or renders obvious, the limitation “determining a number of load and sort processes to be started in parallel based on the identified memory constraints and processing capabilities.”

Hintz and Bordonaro fail to overcome the deficiencies of Bhattacharya. Recall that Hintz was cited only against dependent claims 2-3, and Bordonaro was cited only against dependent claims 4-6. Moreover, Hintz was cited only for determining a number of build processes based on the number of sort processes, and for teaching that the number of sort processes does not exceed a number of indexes to be built, while Bordonaro was cited only for teaching that the number of load processes does not exceed a number of partitions to be loaded, and that the load and sort processes directly dependent on memory constraints. None of these teachings are relevant to the limitations of Applicants’ independent claims.

Thus, Applicants submit that independent claims 1, 20 and 39 are allowable over the references. Further, dependent claims 2-19, 21-38 and 40-57 are submitted to be allowable over the references in the same manner, because they are dependent on independent claims 1 and 12, respectively, and thus contain all the limitations of independent claims 1 and 12. In addition, dependent claims 4-9, 11-25 and 27-44 recite additional novel elements not shown by the references.

V. Conclusion

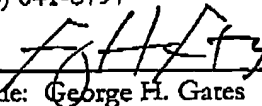
In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited. Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicants' undersigned attorney.

Respectfully submitted,

GATES & COOPER LLP  
Attorneys for Applicants

Howard Hughes Center  
6701 Center Drive West, Suite 1050  
Los Angeles, California 90045  
(310) 641-8797

Date: February 7, 2003

By:   
Name: George H. Gates  
Reg. No.: 33,500

GHG/sjm

G&C 30571.279-US-01